Form Control Arguments:- First argument is reserved for the values

app.component.ts

```
export class AppComponent
{
  title = 'reactiveOnline class';
  form = new FormGroup({
    name : new FormControl("lavanya"),
    email : new FormControll("lavanya@gmail.com"),
    password : new FormControll("pass")
  })
  signUp()
  {
    console.log(this.form)
  }
}
```

Output:-

Name

lavanya

Email

lavanya@gmail.com

Password

pass

Submit

app.component.ts

```
export class AppComponent{
  title = 'reactive Online';
  form = new FormGroup({
  username : new FormControl("", [Validators.required,
  , Validators.minlength(3), Validators.maxlength(15)])
})

  get un()
  {
    return this.form.get('username')
  }

  fun()
  {
    console.log(this.form)
  }
```

app.component.html:

```
<h1> reactive form </h1>
<div class="container">
  <form [formGroup]="form" (ngSubmit)="fun()">
<div class="form-group">
    <lable> Password </lable>
  <input type="text" class="form-control"
        formControlName="username">
<div *ngIf="un.touched && un.invalid">
<div*ngIf="un.error.required > password required </div>
<div *ngIf="un.error.minlength> min length is 3 </div>
<div *ngIf="un.error.maxlength> max length is 15 </div>
</div>
</div>
```

class = "text-danger"

```
<button class="btn btn-primary" type="submit"> Submit
                                              </button>

  </form>
</div>
```

Output:-

Case (1)

Password

[ ]

password is required

[Submit]

Case (2):-

Password

[CH]

min length is 3

[Submit]

Case (3):-

Password

[CHANDUUUUUUUUUU]

max length is 15

[Submit]

* get() returns the particular property (with respect to username we have created on as the reference.
* Template driven form and reactive form both has same methods and properties we are invoking with the help of reference (object)